Defeating MAC Address Randomization Through Timing Attacks

Célestin Matte[†], Mathieu Cunche[†], Franck Rousseau[‡], Mathy Vanhoef[¤]
[†]Univ Lyon, INSA Lyon, Inria, CITI, France,
[‡]Grenoble Alpes, Grenoble Institute of Technology, LIG, France, [¤]iMinds-Distrinet, KU Leuven

ABSTRACT

MAC address randomization is a common privacy protection measure deployed in major operating systems today. It is used to prevent user-tracking with probe requests that are transmitted during IEEE 802.11 network scans. We present an attack to defeat MAC address randomization through observation of the timings of the network scans with an off-the-shelf Wi-Fi interface. This attack relies on a signature based on inter-frame arrival times of probe requests, which is used to group together frames coming from the same device although they use distinct MAC addresses. We propose several distance metrics based on timing and use them together with an incremental learning algorithm in order to group frames. We show that these signatures are consistent over time and can be used as a pseudo-identifier to track devices. Our framework is able to correctly group frames using different MAC addresses but belonging to the same device in up to 75% of the cases. These results show that the timing of 802.11 probe frames can be abused to track individual devices and that address randomization alone is not always enough to protect users against tracking.

CCS Concepts

•**Networks** → **Network privacy and anonymity;** •**Security and privacy** → *Mobile and wireless security;*

Keywords

Security; Privacy; 802.11; Tracking; MAC address randomization

1. INTRODUCTION

Wi-Fi devices are periodically sending frames containing a unique identifier (the MAC address) which can be leveraged to track the owners of those devices. As a countermeasure against tracking, MAC address randomization is becoming an industry standard and is being deployed in most major OSes: iOS since version 8 [9], Windows 10 [10], Android 6.0 [1] and Linux kernel 3.18 [5].

Recently, Vanhoef et al. [10] showed that probe requests contain enough information to form a fingerprint of a device, even without a reliable link-layer identifier. In the present paper, we use even stronger conditions: we suppose that we don't have access to data-link layer information, except the randomized MAC address. In other words, we go further than this previous paper, supposing the previously demonstrated flaws were fixed, and devices stopped adding identifying information in probe requests.

Instead, we study the feasibility of tracking devices based on timing information only. More particularly, we exploit the fact that the frames sent by Wi-Fi devices follow regular patterns that can be used for time-based fingerprinting [3].

The difficulty of such a technique, compared to classical fingerprinting [3], is that we only have a small number of frames to fingerprint a device. In many implementations, random MAC addresses change after a small number of frames have been sent. As a result, we can only gather a small amount of information for each random MAC address. Our solution has to work with this small quantity of information, and has to be reliable at the same time. Besides, as we do not know the number of devices communicating, we cannot build a database of devices before the attack, we have to restrict ourselves to incremental learning methods. Thus, we have to build an hybrid attack that can reliably cluster frames from an unknown number of devices.

This work makes the following contributions. First, we show that the time-based signature is consistent over time, making it a possible pseudo-identifier of a device. Then, we propose an algorithm able to defeat MAC address randomization, in the sense that we are able to group frames from the same device despite the use of random MAC addresses. This attack only relies on timing information and is able to reach an accuracy of up to 75%.

The paper is organized as follows. Section 2 introduces some background information. Section 3 details our algorithm and its various components and parameters. Section 4 presents the results of the tests of our solution, and discusses the efficiency of its parameters. Section 5 concludes the paper.

2. BACKGROUND

2.1 Probe requests

In order to discover 802.11 networks, devices frequently send probe request frames. These frames contain enough

WiSec'16, July 18-22, 2016, Darmstadt, Germany
ACM ISBN 978-1-4503-4270-4/16/07…$15.00
DOI: http://dx.doi.org/10.1145/2939918.2939930

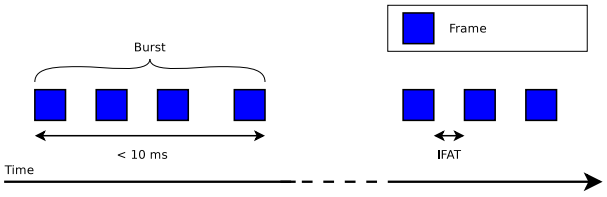

Figure 1: Transmission sequence of frames with Inter-Frame Arrival Time (IFAT) and burst that is a group of frames sent by a device within a time window smaller than 10 ms.

information to identify a device in many cases [10]. For most devices, frames are sent in groups within a small timeframe (less than 10ms), each frame of the group containing a different searched network name (SSID). Such groups of frames are called *bursts* (see Figure 1).

2.2 MAC address randomization schemes

Linux supports MAC address randomization, and lets the driver or firmware generate per-burst random MAC addresses. Indeed, only recent firmware allow for changing the MAC address at each burst but because manufacturers are slow in updating the firmware of Wi-Fi devices, most Linux devices change their MAC address at most every few bursts [5]. As a result, most Linux devices change their MAC address at most every few bursts. The default duration for a random MAC address in `wpa_supplicant` is 60 seconds.

MAC address randomization is supported by iOS since version 8. Randomization is limited to probing and only happens under specific conditions: the device has to be unassociated and in sleep mode [7]. We observe different kinds of behaviour with an iPad 2 running iOS 9.1: the device usually changes its MAC address every few bursts (2-4 bursts), and sometimes changes it for every burst. The conditions for these different behaviors to happen are still to be determined.

Some implementations, such as the one in Windows 10 [6] or the one used in the privacy-oriented Linux distribution Tails do not change the MAC address regularly for Wi-Fi service discovery. Windows 10 changes the MAC address when the device connects or disconnects from a network, and when it is restarted. For such implementations, tracking is trivial since the device identifier does not change during a tracking session.

To sum up, with current implementation of randomization, the same MAC address is used over at least one burst and can cover multiple consecutive bursts.

2.3 Related work

Franklin *et al.* showed that the wireless driver of Wi-Fi-enabled devices can be fingerprinted using the inter-arrival time of the probe requests [3]. They did not discuss the efficiency of their method to distinguish individual devices. As a single driver is used by a great number of devices, being able to distinguish between devices adds a level of precision necessary for tracking.

Freudiger performed an extensive study of Wi-Fi probe requests [4]. The author showed how often certain devices send such frames. He discovered that the number and the frequency of probe requests sent by a device depend on their number of known networks, which hints at the possibility to more precisely fingerprint devices. More specifically, bursts of probe requests are sent with different timings depending on the number of configured networks. He also showed that MAC address randomization can be defeated using sequence numbers.

Wiedersheim *et al.* proposed a method to break pseudonym in Inter-Vehicular Networks [11]. Using a technique called Multiple Hypothesis Tracking, they reached accuracy of almost 100% to track vehicles changing their pseudonym every 10 seconds and sending one beacon message per second. This algorithm makes a strong use of the position of the devices, which is unknown in our case (or at best very imprecise with RSSI), since we only have a single sensor.

Pang *et al.* showed how *implicit identifiers* can be used to track devices and discussed how it rendered pseudonyms insufficient to prevent tracking [8]. They studied several link-layer fields used by associated devices and used a naive Bayes classifier to distinguish between those devices. As we focus on unassociated devices, we do not have access to most of these fields.

2.4 Threat model

We consider an attacker able to monitor the wireless signals in the vicinity of the target, using one off-the-shelf Wi-Fi card. Such an attacker can thus only monitor one channel at a single location. We make this strong assumption so as to show that the location information is not necessary to track devices. This attacker has access to the timing information and the MAC address of each probe request frame, but not more. We make this assumption to consider a situation where the information leakage in Wi-Fi passive discovery has been fixed, as the latter has already been shown to allow device tracking [10].

The goal of this attacker is to distinguish the signals of all devices in range from the crowd even though they use MAC address randomization, and to track individual devices among extended periods of time.

This simple attacker model can be further extended by considering advanced techniques and using several sensors. Adding the location information obtained by different sensors would improve the accuracy of our algorithm.

This attacker model is complementary to the one used in [11], which focuses on location information to track vehicles using pseudonyms. In fact, both attacks could be combined to defeat pseudonymization.

3. DEFEATING RANDOMIZATION USING TIMING

We present an attack grouping probe requests together based on the sending device despite the use of a changing link-layer identifier. To do so, we use a timing-based method, which considers inter-frame arrival time (IFAT) between frames using the same MAC address. We compute signatures for each group of frames using the same MAC address and compare these signatures using custom distances.

3.1 Terminology

We introduce the following definitions:

- a *burst* is a group of probe request frames sent by a device within 10ms,
- a *burst set* is a group of bursts sent with the same (possibly random) MAC address,
- an *alias* of a MAC address is another (random) MAC address used by the same device,

Algorithm 1: Random MAC breaking

Input: $\mathcal{G}$: groups of burst sets, grouped by MAC address
t: distance threshold
d: a distance function
Returns: $\mathcal{A}$: dictionary of aliases

$\mathcal{A} \leftarrow \emptyset$
$\mathcal{D} \leftarrow \emptyset$ `// Database of signatures`
foreach $\mathcal{B} \in \mathcal{G}$ **do**
  $\mathcal{S} \leftarrow$ signature($\mathcal{B}$)
  $d_{min} \leftarrow min(d(\mathcal{S}, \mathcal{S}')$ where $\mathcal{S}' \in \mathcal{D})$
  if $d_{min} < t$ **then**
    $\mathcal{A}[\mathcal{B}.mac] \leftarrow \mathcal{A}[\mathcal{S}'.mac]$
  else
    $\mathcal{A}[\mathcal{B}.mac] \leftarrow \mathcal{B}.mac$
  end
  $D \leftarrow \mathcal{D} \cup \mathcal{S}$
end
return $\mathcal{A}$

- Inter-Frame Arrival Time (IFAT) is the time difference between two frames.

Also, *randomization* will be used as short for *MAC address randomization* throughout the paper.

3.2 Frame grouping algorithm

Our algorithm takes as input a capture of probe requests and outputs a mapping between the frames and a set of identifiers. Ideally, each identifier should correspond to a distinct device and its associated frames. The mapping is obtained by grouping together frames that appear to originate from the same device based on timing information. More particularly, our algorithm relies on timing-based distances and on an incremental learning algorithm customized to fit the constraints of our use case.

We build a database of time-based signatures for the different MAC addresses, as described in [3]. We divide time into discrete timeframes of equal size (bins). For each group G of frames using the same MAC address, we calculate the inter-frame arrival time (IFAT) between each pair of consecutive frames. We then calculate the ratio and mean value of IFATs in each bin, which constitutes the signature $\mathcal{S}(G)$ for group G. For G, let P_b^G be the percentage of frames in a bin b, and M_b^G the mean IFAT value in bin b. Let $\mathcal{B}$ be the set of all possible bins, the signature $\mathcal{S}$ of group G is given by:

$$\mathcal{S}(G) = \{P_b^G, M_b^G | b \in \mathcal{B}\}$$

For each burst set, we calculate the distance between the signature of this group and every other known signature. If at least one of these distances is below a given threshold t, we choose the MAC address of the signature yielding minimal distance, and consider the two MAC addresses to belong to the same device. Otherwise, we estimate the MAC address to belong to a new device. We add the signature of the new burst set to the database.

We consider two options for this algorithm:

- online: try to group burst sets with previous burst sets only (as described in Algorithm 1).
- offline: try to group burst sets with any burst set

3.3 Distance

The previous algorithm relies on a distance metric in order to group frame together. In this section, we introduce several timing-based distance metrics derived from the one originally introduced by Franklin *et al.* [3].

3.3.1 Franklin's distance

The first considered distance is a modification of the one used by Franklin to fingerprint device drivers. We modify Franklin's distance formula so that it respects the symmetric propriety of a distance. Instead of multiplying the difference of the means by the percentage of a single device, we multiply by the mean of the percentages of both devices.

The distance between two burst sets A and B is based on their signatures. We calculate the distance $D1_{AB}$ using the following formula:

$$D1_{AB} = \sum\nolimits_{b \in \mathcal{B}} (|P_b^B - P_b^A| + \frac{(P_b^A + P_b^B)}{2} * |M_b^B - M_b^A|)$$

Percentages and means are set to 0 if the bin is empty.

As our distance uses inter-frame arrival time, single frames cannot be considered. We choose to ignore them.

3.3.2 Adding inter-burst set arrival time

As opposed to previous work, the number of frames on which the fingerprint can be computed is limited to a small number of bursts because the MAC address is changed periodically (see section 2). To deal with this reduced amount of information, we extend the previous distance by considering inter-arrival time between the compared burst. The underlying assumption is that bursts exhibit a temporal regularity.

Assuming a group B is composed of frames seen later than those of group A, we calculate the IFAT between the last frame of A and the first one of B: $\text{IFAT}_{AB} = B.first - A.last$. We then check if this IFAT exists in the signature $\mathcal{S}(A)$. If this is not the case, we consider the distance to be $+\infty$. Otherwise, we look at the percentage p of the signature of the IFAT's bin. We multiply the distance by $1 - p$.

So, if we call $D2_{AB}$ this new distance:

$$D2_{AB} = \begin{cases} +\infty \text{ if } \text{IFAT}_{\text{AB}} \notin S_A \\ (1 - P_b^A) * D1_{AB} \text{ otherwise} \end{cases}$$

In our algorithm, two burst sets with infinite distance will never be grouped as they are assumed to have very few chance to come from the same device.

3.3.3 Hybrid distance

Because of frame losses and delays in burst transmissions, the previous distance metric may lead to a large amount of false negatives. Therefore, we introduce $D3$, which constitutes a trade-off between $D1$ and $D2$: instead of giving an infinite value, we multiply $D1$ by a constant C (the choice of the value of this constant is discussed in section 4.8). In other words:

$$D3_{AB} = \begin{cases} C * D1_{AB} \text{ if } \text{IFAT}_{\text{AB}} \notin S_A \\ (1 - P_b^A) * D1_{AB} \text{ otherwise} \end{cases}$$

As a result, this distance favors groups of frames having a coherent IFAT with the current group, but seeks to group frames even if no group with coherent IFAT is found.

3.4 Knowledge-keeping algorithm

Our algorithm groups burst sets together, and stores these relationships in an internal structure. It does not use previous knowledge about aliases to group new-coming burst sets, so as not to spread error to further grouping guesses. In concrete terms, a new burst set is compared to each previous burst set. It can thus be compared to several burst sets from a device.

Another way to proceed would be to use previous knowledge of which burst sets are assumed to belong to the same device by creating meta-groups of frames. Signatures of these meta-groups would include more frames, making the results possibly more accurate. A new burst set is then compared once to each known device.

3.5 Nearest neighbors method

We consider an improvement of our algorithm by borrowing the approach of the k-nearest neighbors (k-NN) algorithm. When trying to group a burst set S with other burst sets, we compute the distances with all other burst sets. Instead of grouping S with the burst sets which yields the lowest distance, we have a look at the k burst sets which yield the lowest distances, and group S with the burst set which is most present within those k burst sets.

With this modified algorithm, groups of frames from random MAC addresses will score a low distance with all the previous groups from the same device, and should then have a better chance to be classified correctly.

4. EXPERIMENTS AND RESULTS

We first evaluate our distances in order to find the most efficient one to estimate if two groups of frames come from the same device or from different ones. We then proceed to improve the efficiency of our solution by overviewing the impact of the different parameters: distance threshold, size of the bins for the distance.

4.1 Dataset

In order to estimate our solution, we use a real-world dataset free of random MAC addresses. This dataset of more than 120 000 probe requests sent by over 550 devices was collected in our laboratory, over a period of 6 days.

We transform it into a dataset containing devices changing their MAC addresses every p bursts of probe requests. For evaluation purpose, the resulting dataset keeps a trace of which probe requests come from the same device despite MAC address randomization. This allows us to keep the ground truth data.

We generate a trace having 100 among the 550 devices using random MAC addresses for burst sets of 4 bursts.

4.2 Distance metric evaluation

In order to evaluate the efficiency of our distance, we calculate the distances between burst sets from the same devices, as well as the distances between burst sets from different devices. We then compare those two groups of distances in order to see if our distance metric can be used to identify if two burst sets come from the same device or from different ones. We only do this for $D1$, as $D2$ and $D3$ will fail to evaluate non-consecutive burst sets properly since they use inter-burst arrival time.

Results can be seen in Figure 2. We observe that bursts from distinct devices have a larger distance than the bursts from the same device, which confirms the utility of this distance for our attack.

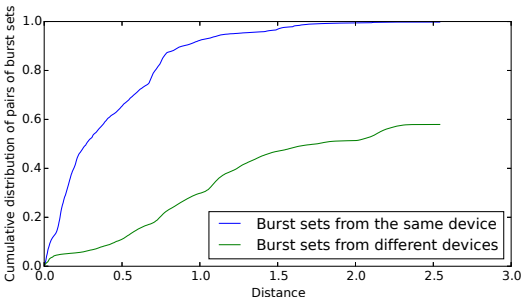

Figure 2: Cumulative distribution of the distance $D1$ for pairs of burst set from the same device and from distinct devices.

4.3 Stability of the distance over time

In this section, we study the stability of the considered distance over time in order to confirm that our algorithm can work to track a device over an extended period of time. To do so, we compute the distance between groups of frames sent by the same device at distinct time intervals.

We consider several time differences between 1 minute and up to 50 days. For each of them, we select 50 devices for which frames separated by this time difference ($\pm 10\%$) are available in our dataset. We take two groups of 1 minute of these frames at each extremity of the time difference and compute the distance between these groups. In order to have data covering an extended time period, we use the Sapienza dataset [2] as it covers 50 days, a longer time period than our own. Thanks to this dataset, we can calculate the consistency of the distance over long time periods.

The results are shown on figure 3. We observe an increase of less than 50% even after 50 days, which seems low enough to allow tracking devices over a long time period. Values reached by the distance even after 50 days still make a difference of 49% between the two kinds of burst sets in figure 2 (42% in the worst case), while the average distance for burst sets with 1 minute of time interval make a difference of 61%.

In our dataset and experimental conditions, we do not observe the probable IFAT deviation mentioned by Freudiger [4], observed when disabling Google services on Android.

4.4 Performance metrics

We evaluate the performances of the proposed algorithm based on a number of metrics: accuracy, true positive rate and false positive rate.

We define the accuracy metrics as being the ratio of correct decisions made by our algorithm. We define a correct decision as being either:

- a burst set using a random MAC address correctly grouped with another burst set from the same device;
- a burst set from a device not using a random MAC address not grouped with any other burst sets (i.e. a group of frames from a device using a normal MAC address is not grouped with frames from another device).

Accuracy in itself does not provide information about the

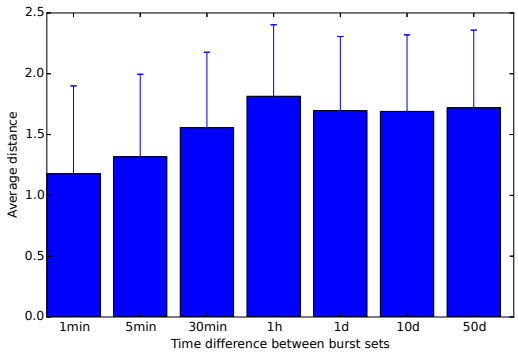

Figure 3: mean and standard deviation of the distance w.r.t. time difference between groups of frames.

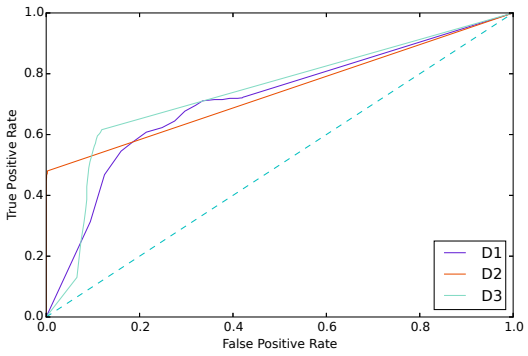

Figure 4: ROC curve of the three distances, over the range of threshold values.

kinds of errors made by the algorithm. For that purpose, we define two supplementary metrics:

- True positive rate (TPR): number of burst sets from devices using random MAC addresses correctly grouped together, over the number of burst sets from devices using random MAC addresses,
- False positive rate (FPR): number of burst sets incorrectly grouped with burst sets from other devices, over the total number of burst sets.

TPR gives a better insight about positive results related to devices using random MAC addresses only. FPR shows the ratio of undesirable errors made by the algorithm (burst sets from different devices grouped together). In an attack, a false positive may lead to a device being taken for another one. As a result, the actual device may not be detected, while the other device may be missed. Nevertheless, such errors will only be significant if they appear on a long number of successive burst sets, as they could be detected by human judgement or a filtering algorithm otherwise.

4.5 Performances as a function of the threshold

We plot the performance of our algorithm with the 3 distances as a function of the distance threshold t, summarized in a ROC curve (Figure 4). With distance $D1$, we observe a maximum for the accuracy metrics for $t = 0.4$, with accuracy $= 62.7\%$. Increasing the distance threshold then increases both true positive and false positive rates.

With distance $D2$, our false positive rate stays close to 0. Thanks to this, we can afford higher values of t, for which the FPR stays much smaller than 1%. However, our TPR is not as good as with distance $D1$, and our algorithm fails to find the original MAC address for more than half groups of frames using random MAC addresses. With distance $D1$, we could succeed for more than 70% of such groups, at a cost of a high FPR.

Further investigations show that higher values of t (such as 10) do not modify the results anymore. As a result, with distance $D2$, the value of t can be taken arbitrarily high for optimal results.

As expected, distance $D3$ appears to be a good trade-off between $D1$ and $D2$: results values for TPR and FPR lie between those of $D1$ and $D2$.

4.6 Knowledge-keeping algorithm evaluation

We evaluate our knowledge-keeping algorithm (see section 3.4) on our 3 distances.

As this method is more prone to error accumulation, it is not surprising that the best results are given by distance $D2$, which yields almost no false positive. Distance $D3$ becomes victim of this error accumulation for high values of t, due to a higher FPR, and sees an accuracy drop of about 8%. Performances for distance $D1$ are definitely worse, with a fall in accuracy of about 18%. Performances for distance $D2$ are almost equal with both strategies (the difference is less than 0.01%).

To sum up, the knowledge-keeping algorithm does not improve performances but has the potential to make them drop in some cases.

4.7 Influence of the parameters

In order to improve the results of the algorithm, we evaluate the influence of several parameters on its efficiency: size of the bins, number of future bursts.

4.7.1 Temporal granularity/Size of the bins

To build our signatures for the MAC addresses, we discretize time by forming bins of fixed size. We vary the size of these bins used to calculate the distance between groups of frames, from $1\mu s$ to $1s$. What appears is that the size of the bins does not matter much, as long as it is bigger than a threshold of $50ms$ to $300ms$, depending on the distance used. bins smaller than this threshold result in a lower accuracy. The best results are obtained with bins of size $670ms$ which an accuracy of 76.2%. Tests for higher values (1 to 5s) do not yield better results.

4.7.2 Number of future bursts

We attempt to use our algorithm with harder conditions: instead of considering all frames with the same MAC address to build signatures, we only take the first N bursts in time, with N from 1 to 5. The aim of this harder condition is to see if our algorithm can be used in near real-time, where the amount of information is reduced to a small timeframe.

Results are presented in Figure 5. Not fetching bursts of probe requests in the future yields a drop of 40% in the accuracy of our algorithm. Actually, no probe request with a random MAC address is classified correctly ($TPR = 0$). The

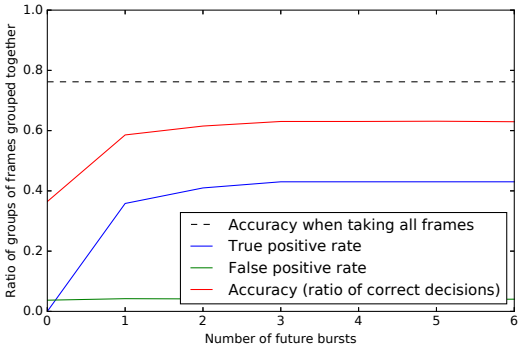

Figure 5: Influence of the number of future bursts fetched for distance $D2$.

Table 1: Results of the attack with the best parameters and options.

Distance	Accuracy	TPR	FPR
D1	66.8%	74.1%	24.3%
D2	**77.2%**	64.0%	**0.6%**
D3	71.8%	**75.2%**	17.5%

accuracy of 40% is only due to probe requests not using random MAC addresses not being grouped with other frames. This shows that our algorithm fails to classify groups of only one burst of frames. With two groups of frames or more, results become usable with a drop in accuracy of less than 20%. Above 4 groups of bursts, results don't vary since we simulate devices using random MAC addresses by randomizing groups of 4 bursts.

4.8 Other parameters and options

This section ends the discussion about the tested parameters. We evaluate the effectiveness of the two options presented in section 3.2. We observe an increase of 2 to 3% for the 3 distances with the offline algorithm.

We evaluate our k-NN method (see section 3.5) with the 3 distances. 1-NN corresponds to the default behaviour, where we only consider the closest group of frames. Results show that this method is not more efficient for any value of k greater than 1: tests on the values of k from 2 to 10 on the 3 distances yield a ratio decrease of 1 to 30%.

For the choice of the constant used for distance $D3$, we calculate accuracy, TPR and FPR for various values of this constant, ranging from 1 to 100. 10 appears to be the best choice for this constant as it maximizes accuracy.

4.9 Results summary

Our estimated best parameters are: $t = 0.4$ for distance $D1$, $t = 2.0$ for distances $D2$ and $D3$, bins of size 670ms, offline algorithm, not using the knowledge-keeping algorithm. With these parameters, we obtain results presented in Table 1. $D2$ appears to yield both the best accuracy and the lowest FPR, whereas $D3$ yields the best TPR. Depending on the conditions, $D2$ or $D3$ may be considered the best distance metric.

5. CONCLUSION

In this work, we presented an attack capable of tracking Wi-Fi devices over time despite the use of MAC address randomization mechanisms. Our timing-based attack is able to successfully group together frames originating from the same device, but having a distinct MAC address pseudonym in 77.2% of the cases.

This new class of attack shows that owners of Wi-Fi devices are exposed to tracking. Furthermore, it demonstrates that the content of Wi-Fi frames is not even necessary in order to track devices.

Based on our observations and the results of our experiments, we can devise several countermeasures that would reduce the effectiveness of the attack presented in this paper. First, changing the MAC address more often, e.g. every burst or every frame, has the potential to reduce the trackability of devices, as the amount of information for fingerprinting will be limited to a few frames. Then, since our attack relies on temporal pattern, a simple countermeasure would be to break those patterns by introducing some random delays between probes and between bursts.

6. REFERENCES

[1] Android 6.0 changes. Retrieved from https://developer.android.com/about/versions/marshmallow/android-6.0-changes.html, 2015.

[2] M. V. Barbera, A. Epasto, A. Mei, S. Kosta, V. C. Perta, and J. Stefa. CRAWDAD dataset sapienza/probe-requests (v. 2013-09-10). Retrieved 10 November, 2015, from, http://crawdad.org/sapienza/probe-requests/20130910, Sept. 2013.

[3] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. V. Randwyk, and D. Sicker. Passive data link layer 802.11 wireless device driver fingerprinting. In *Usenix Security*, volume 6, 2006.

[4] J. Freudiger. How talkative is your mobile device?: an experimental study of Wi-Fi probe requests. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 2015.

[5] E. Grumbach. iwlwifi: mvm: support random MAC address for scanning. Linux commit `effd05ac479b`.

[6] C. Huitema. Experience with mac address randomization in windows 10, 2015.

[7] B. Misra. ios8 mac randomization – analyzed! http://blog.mojonetworks.com/ios8-mac-randomization-analyzed/, 2014.

[8] J. Pang, B. Greenstein, R. Gummadi, S. Seshan, and D. Wetherall. 802.11 user fingerprinting. In *MobiCom*, pages 99–110. ACM, 2007.

[9] K. Skinner and J. Novak. Privacy and your app. In *Apple Worldwide Dev. Conf. (WWDC)*, June 2015.

[10] M. Vanhoef, C. Matte, M. Cunche, L. Cardoso, and F. Piessens. Why MAC Address Randomization is not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms. In *AsiaCCS*, May 2016.

[11] B. Wiedersheim, Z. Ma, F. Kargl, and P. Papadimitratos. Privacy in inter-vehicular networks: Why simple pseudonym change is not enough. In *Wireless On-demand Network Systems and Services (WONS)*, pages 176–183. IEEE, 2010.